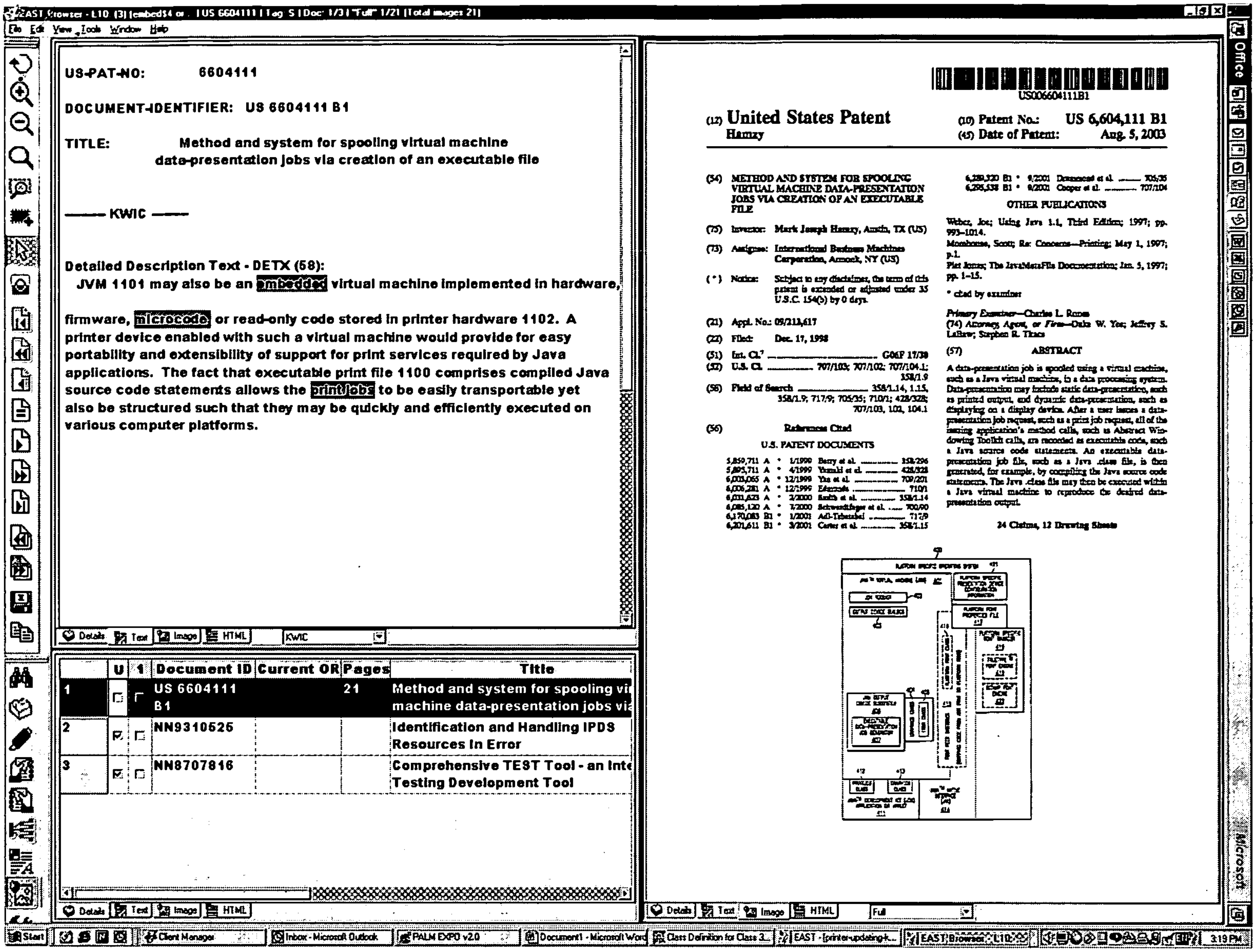

EAST Browser - L10: (3) (embed$4 or . | US 6604111 | Tag: S | Doc: 1/3 | "Full" 1/21 | Total images 21|
File Edit View Tools Window Help
Details Text Image HTML KWIC
Details Text Image HTML
Details Text Image HTML Full
Start Client Manager Inbox - Microsoft Outlook PALM EXPO v2.0 Document1 - Microsoft Word Class Definition for Class 3... EAST - [printer-updating-k... EAST Browser - L10:... 3:19 PM
Office
Microsoft

US-PAT-NO: 6604111

DOCUMENT-IDENTIFIER: US 6604111 B1

TITLE: Method and system for spooling virtual machine data-presentation jobs via creation of an executable file

——— KWIC ———

Detailed Description Text - DETX (58):

JVM 1101 may also be an embedded virtual machine implemented in hardware, firmware, microcode, or read-only code stored in printer hardware 1102. A printer device enabled with such a virtual machine would provide for easy portability and extensibility of support for print services required by Java applications. The fact that executable print file 1100 comprises compiled Java source code statements allows the printjobs to be easily transportable yet also be structured such that they may be quickly and efficiently executed on various computer platforms.

	U	1	Document ID	Current OR	Pages	Title
1	□	□	US 6604111 B1		21	Method and system for spooling vi machine data-presentation jobs vi
2	☑	□	NN9310526			Identification and Handling IPDS Resources In Error
3	☑	□	NN8707816			Comprehensive TEST Tool - an Inte Testing Development Tool

US006604111B1

(12) United States Patent
Hamzy

(10) Patent No.: US 6,604,111 B1
(45) Date of Patent: Aug. 5, 2003

(54) METHOD AND SYSTEM FOR SPOOLING VIRTUAL MACHINE DATA-PRESENTATION JOBS VIA CREATION OF AN EXECUTABLE FILE

(75) Inventor: Mark Joseph Hamzy, Austin, TX (US)

(73) Assignee: International Business Machines Corporation, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: 09/213,617

(22) Filed: Dec. 17, 1998

(51) Int. Cl.^{7} G06F 17/30

(52) U.S. Cl. 707/103; 707/102; 707/104.1; 358/1.9

(58) Field of Search 358/1.14, 1.15, 358/1.9; 717/9; 705/35; 710/1; 428/328; 707/103, 102, 104.1

(56) References Cited

U.S. PATENT DOCUMENTS

5,859,711	A	*	1/1999	Barry et al.	358/296
5,895,711	A	*	4/1999	Yamaki et al.	428/328
6,003,065	A	*	12/1999	Yan et al.	709/201
6,006,281	A	*	12/1999	Edmunds	710/1
6,031,623	A	*	2/2000	Smith et al.	358/1.14
6,085,120	A	*	7/2000	Schwerdtfeger et al.	700/90
6,170,083	B1	*	1/2001	Adl-Tabatabai	717/9
6,201,611	B1	*	3/2001	Carter et al.	358/1.15
6,289,320	B1	*	9/2001	Drummond et al.	705/35
6,295,538	B1	*	9/2001	Cooper et al.	707/104

OTHER PUBLICATIONS

Weber, Joe; Using Java 1.1, Third Edition; 1997; pp. 993–1014.

Mombourne, Scott; Re: Concerns—Printing; May 1, 1997; p.1.

Piet Jonas; The JavaMetaFile Documentation; Jan. 5, 1997; pp. 1–15.

* cited by examiner

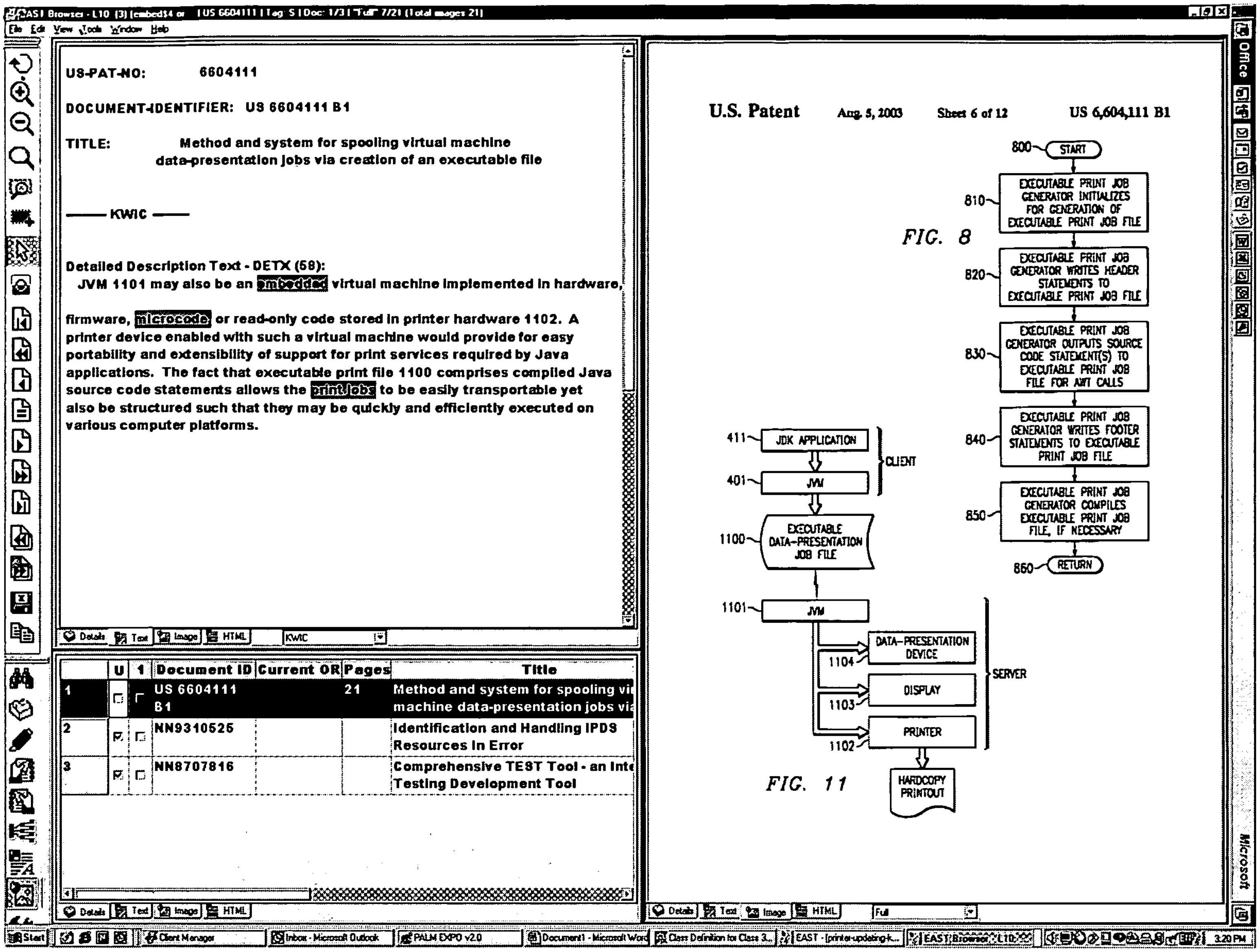
Primary Examiner—Charles L. Rones

(74) *Attorney, Agent, or Firm*—Duke W. Yee; Jeffrey S. LaBaw; Stephen R. Tkacs

(57) ABSTRACT

A data-presentation job is spooled using a virtual machine, such as a Java virtual machine, in a data processing system. Data-presentation may include static data-presentation, such as printed output, and dynamic data-presentation, such as displaying on a display device. After a user issues a data-presentation job request, such as a print job request, all of the issuing application's method calls, such as Abstract Windowing Toolkit calls, are recorded as executable code, such a Java source code statements. An executable data-presentation job file, such as a Java .class file, is then generated, for example, by compiling the Java source code statements. The Java .class file may then be executed within a Java virtual machine to reproduce the desired data-presentation output.

24 Claims, 12 Drawing Sheets



EAST Browser - L10 (3) (embed$4 or | US 6604111 | Tag: S | Doc: 1/3 | "Full" 7/21 (Total images 21)
File Edit View Tools Window Help

US-PAT-NO: 6604111

DOCUMENT-IDENTIFIER: US 6604111 B1

TITLE: Method and system for spooling virtual machine data-presentation jobs via creation of an executable file

—— KWIC ——

Detailed Description Text - DETX (58):

JVM 1101 may also be an embedded virtual machine implemented in hardware, firmware, microcode, or read-only code stored in printer hardware 1102. A printer device enabled with such a virtual machine would provide for easy portability and extensibility of support for print services required by Java applications. The fact that executable print file 1100 comprises compiled Java source code statements allows the print jobs to be easily transportable yet also be structured such that they may be quickly and efficiently executed on various computer platforms.

	U	1	Document ID	Current OR	Pages	Title
1	□	□	US 6604111 B1		21	Method and system for spooling vi machine data-presentation jobs vi
2	☑	□	NN9310525			Identification and Handling IPDS Resources In Error
3	☑	□	NN8707816			Comprehensive TEST Tool - an Inte Testing Development Tool

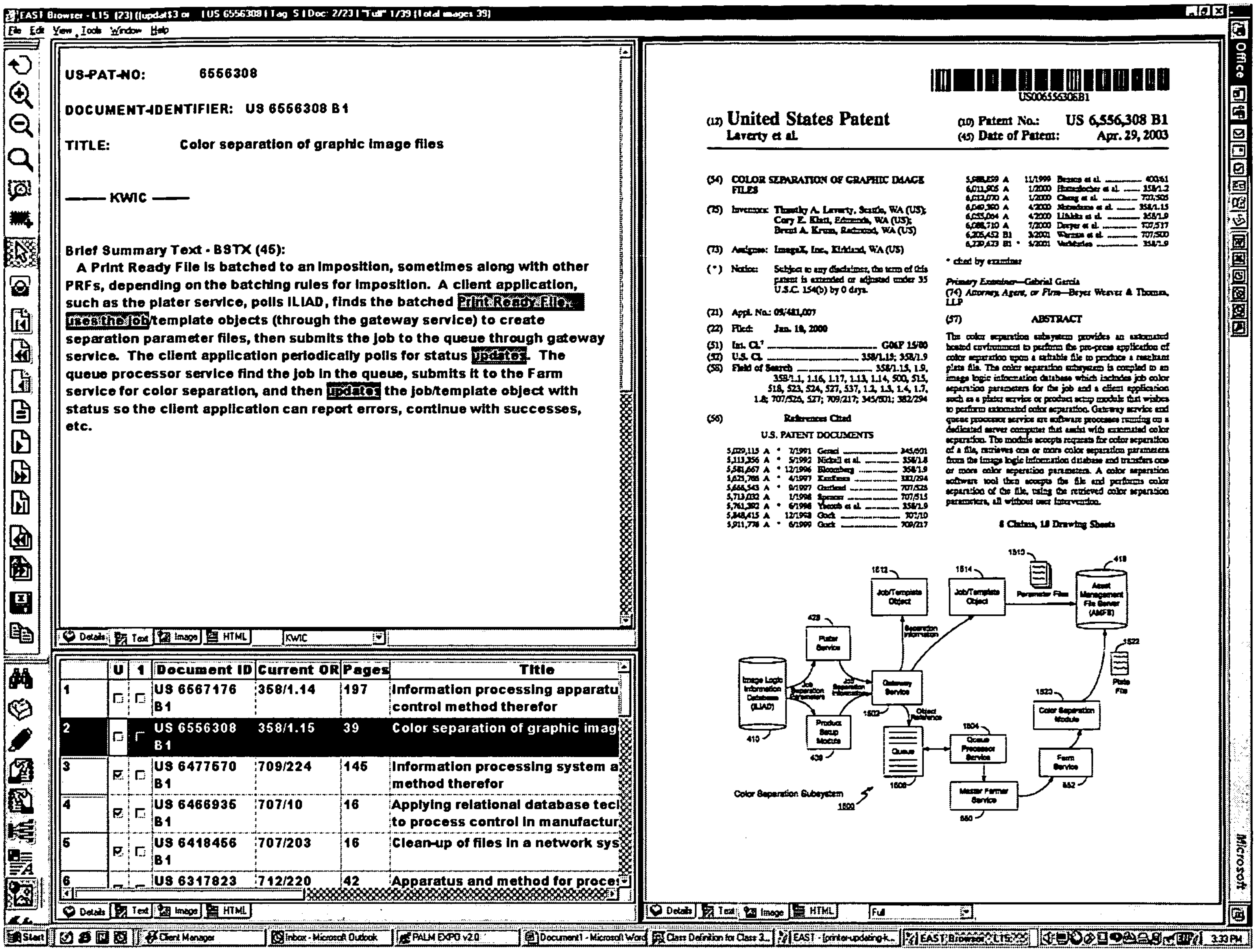
U.S. Patent Aug. 5, 2003 Sheet 6 of 12 US 6,604,111 B1

800
START
810
EXECUTABLE PRINT JOB GENERATOR INITIALIZES FOR GENERATION OF EXECUTABLE PRINT JOB FILE
820
EXECUTABLE PRINT JOB GENERATOR WRITES HEADER STATEMENTS TO EXECUTABLE PRINT JOB FILE
830
EXECUTABLE PRINT JOB GENERATOR OUTPUTS SOURCE CODE STATEMENT(S) TO EXECUTABLE PRINT JOB FILE FOR AWT CALLS
840
EXECUTABLE PRINT JOB GENERATOR WRITES FOOTER STATEMENTS TO EXECUTABLE PRINT JOB FILE
850
EXECUTABLE PRINT JOB GENERATOR COMPILES EXECUTABLE PRINT JOB FILE, IF NECESSARY
860
RETURN
411
JDK APPLICATION
401
JVM
CLIENT
1100
EXECUTABLE DATA-PRESENTATION JOB FILE
1101
JVM
DATA-PRESENTATION DEVICE
1104
DISPLAY
1103
PRINTER
1102
SERVER
HARDCOPY PRINTOUT

FIG. 8

FIG. 11

Details Text Image HTML Full
Office
Microsoft
Start
Client Manager
Inbox - Microsoft Outlook
PALM EXPO v2.0
Document1 - Microsoft Word
Class Definition for Class 3...
EAST - [printer-updating-k...
EAST Browser - L10
3:20 PM


EAST Browser - L15 (23) ((updat$3 or | US 6556308 | Tag S | Doc: 2/23 | "Full" 1/39 (Total images 39)
File Edit View Tools Window Help
Details Text Image HTML
KWIC
Full
Start
Client Manager
Inbox - Microsoft Outlook
PALM EXPO v2.0
Document1 - Microsoft Word
Class Definition for Class 3...
EAST - [printer-updating-k...
EAST Browser - L15: (23)
3:33 PM
Office
Microsoft

US-PAT-NO: 6556308

DOCUMENT-IDENTIFIER: US 6556308 B1

TITLE: Color separation of graphic image files

——— KWIC ———

Brief Summary Text - BSTX (45):

A Print Ready File is batched to an imposition, sometimes along with other PRFs, depending on the batching rules for imposition. A client application, such as the plater service, polls ILIAD, finds the batched Print Ready File, uses the job/template objects (through the gateway service) to create separation parameter files, then submits the job to the queue through gateway service. The client application periodically polls for status updates. The queue processor service find the job in the queue, submits it to the Farm service for color separation, and then updates the job/template object with status so the client application can report errors, continue with successes, etc.

	U	1	Document ID	Current OR	Pages	Title
1	□	□	US 6567176 B1	358/1.14	197	Information processing apparatu control method therefor
2	□	□	US 6556308 B1	358/1.15	39	Color separation of graphic imag
3	☑	□	US 6477570 B1	709/224	146	Information processing system a method therefor
4	☑	□	US 6466936 B1	707/10	16	Applying relational database tecl to process control in manufactur
5	☑	□	US 6418456 B1	707/203	16	Clean-up of files in a network sys
6			US 6317823	712/220	42	Apparatus and method for proce

US006556308B1

(12) United States Patent
Laverty et al.

(10) Patent No.: US 6,556,308 B1
(45) Date of Patent: Apr. 29, 2003

(54) COLOR SEPARATION OF GRAPHIC IMAGE FILES

(75) Inventors: Timothy A. Laverty, Seattle, WA (US); Cory E. Klatt, Edmonds, WA (US); Brent A. Kruse, Redmond, WA (US)

(73) Assignee: ImageX, Inc., Kirkland, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: 09/481,007

(22) Filed: Jan. 10, 2000

(51) Int. Cl.^{7} G06F 15/00

(52) U.S. Cl. 358/1.15; 358/1.9

(58) Field of Search 358/1.15, 1.9, 358/1.1, 1.16, 1.17, 1.13, 1.14, 500, 515, 518, 523, 524, 527, 537, 1.2, 1.3, 1.4, 1.7, 1.8; 707/526, 527; 709/217; 345/501; 382/294

(56) References Cited

U.S. PATENT DOCUMENTS

5,029,115	A	*	7/1991	Geraci	345/501
5,113,356	A	*	5/1992	Nickell et al.	358/1.8
5,581,667	A	*	12/1996	Bloomberg	358/1.9
5,625,766	A	*	4/1997	Kauffman	382/294
5,666,543	A	*	9/1997	Gartland	707/526
5,713,032	A		1/1998	Spencer	707/515
5,761,392	A	*	6/1998	Yacoub et al.	358/1.9
5,848,415	A		12/1998	Guck	707/10
5,911,776	A	*	6/1999	Guck	709/217
5,988,899	A		11/1999	Benson et al.	400/61
6,011,905	A		1/2000	Huttenlocher et al.	358/1.2
6,012,070	A		1/2000	Cheng et al.	707/505
6,049,390	A		4/2000	Notredame et al.	358/1.15
6,055,064	A		4/2000	Lifshitz et al.	358/1.9
6,088,710	A		7/2000	Dreyer et al.	707/517
6,205,452	B1		3/2001	Warmus et al.	707/500
6,229,623	B1	*	5/2001	Velikides	358/1.9

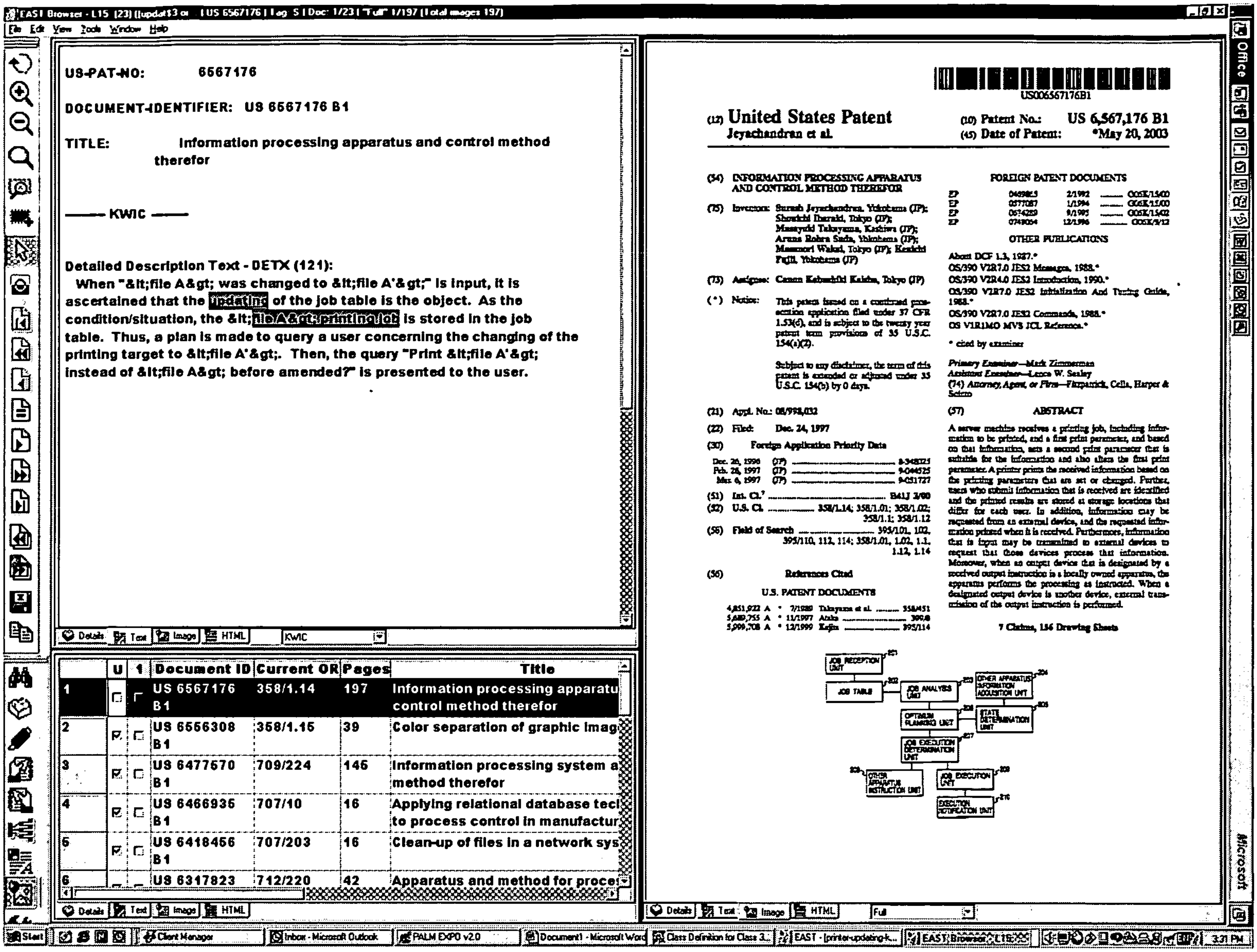
* cited by examiner

Primary Examiner—Gabriel Garcia
(74) *Attorney, Agent, or Firm*—Beyer Weaver & Thomas, LLP

(57) ABSTRACT

The color separation subsystem provides an automated hosted environment to perform the pre-press application of color separation upon a suitable file to produce a resultant plate file. The color separation subsystem is coupled to an image logic information database which includes job color separation parameters for the job and a client application such as a plater service or product setup module that wishes to perform automated color separation. Gateway service and queue processor service are software processes running on a dedicated server computer that assist with automated color separation. The module accepts requests for color separation of a file, retrieves one or more color separation parameters from the image logic information database and transfers one or more color separation parameters. A color separation software tool then accepts the file and performs color separation of the file, using the retrieved color separation parameters, all without user intervention.

8 Claims, 18 Drawing Sheets


EAST Browser - L15 (23) (updat$3 or | US 6567176 | Tag: S | Doc: 1/23 | "Full" 1/197 | Total images 197)
File Edit View Tools Window Help
Details Text Image HTML
KWIC
Full
Office
Microsoft
Start
Client Manager
Inbox - Microsoft Outlook
PALM EXPO v2.0
Document1 - Microsoft Word
Class Definition for Class 3...
EAST - [printer-updating-k...
EAST Browser - L15
3:31 PM

US-PAT-NO: 6567176

DOCUMENT-IDENTIFIER: US 6567176 B1

TITLE: Information processing apparatus and control method therefor

——— KWIC ———

Detailed Description Text - DETX (121):

When "<file A> was changed to <file A'>" is input, it is ascertained that the updating of the job table is the object. As the condition/situation, the <file A> printing job is stored in the job table. Thus, a plan is made to query a user concerning the changing of the printing target to <file A'>. Then, the query "Print <file A'> instead of <file A> before amended?" is presented to the user.

	U	1	Document ID	Current OR	Pages	Title
1	□	□	US 6567176 B1	358/1.14	197	Information processing apparatu control method therefor
2	☑	□	US 6556308 B1	358/1.15	39	Color separation of graphic imag
3	☑	□	US 6477570 B1	709/224	146	Information processing system a method therefor
4	☑	□	US 6466936 B1	707/10	16	Applying relational database tec to process control in manufactur
5	☑	□	US 6418456 B1	707/203	16	Clean-up of files in a network sys
6			US 6317823	712/220	42	Apparatus and method for proce

US006567176B1

(12) **United States Patent**
Jeyachandran et al.

(10) Patent No.: **US 6,567,176 B1**
(45) Date of Patent: ***May 20, 2003**

(54) INFORMATION PROCESSING APPARATUS AND CONTROL METHOD THEREFOR

(75) Inventors: Suresh Jeyachandran, Yokohama (JP); Shoichi Ibaraki, Tokyo (JP); Masayuki Takayama, Kashiwa (JP); Aruna Rohra Suda, Yokohama (JP); Masanori Wakai, Tokyo (JP); Kenichi Fujii, Yokohama (JP)

(73) Assignee: Canon Kabushiki Kaisha, Tokyo (JP)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: 08/998,032

(22) Filed: Dec. 24, 1997

(30) Foreign Application Priority Data

Dec. 26, 1996	(JP)	8-348025
Feb. 28, 1997	(JP)	9-044525
Mar. 6, 1997	(JP)	9-051727

(51) Int. Cl.^{7} B41J 3/00

(52) U.S. Cl. 358/1.14; 358/1.01; 358/1.02; 358/1.1; 358/1.12

(58) Field of Search 395/101, 102, 395/110, 112, 114; 358/1.01, 1.02, 1.1, 1.12, 1.14

(56) References Cited

U.S. PATENT DOCUMENTS

4,851,922 A	* 7/1989	Takayama et al.	....	358/451
5,689,755 A	* 11/1997	Ataka	....	399/8
5,999,708 A	* 12/1999	Kajita	....	395/114

FOREIGN PATENT DOCUMENTS

EP	0469813	2/1992	G06K/15/00
EP	0577087	1/1994	G06K/15/00
EP	0674259	9/1995	G06K/15/02
EP	0749064	12/1996	G06K/9/12

OTHER PUBLICATIONS

About DCF 1.3, 1987.*
OS/390 V2R7.0 JES2 Messages, 1988.*
OS/390 V2R4.0 JES2 Introduction, 1990.*
OS/390 V2R7.0 JES2 Initialization And Tuning Guide, 1988.*
OS/390 V2R7.0 JES2 Commands, 1988.*
OS V1R1MO MVS JCL Reference.*

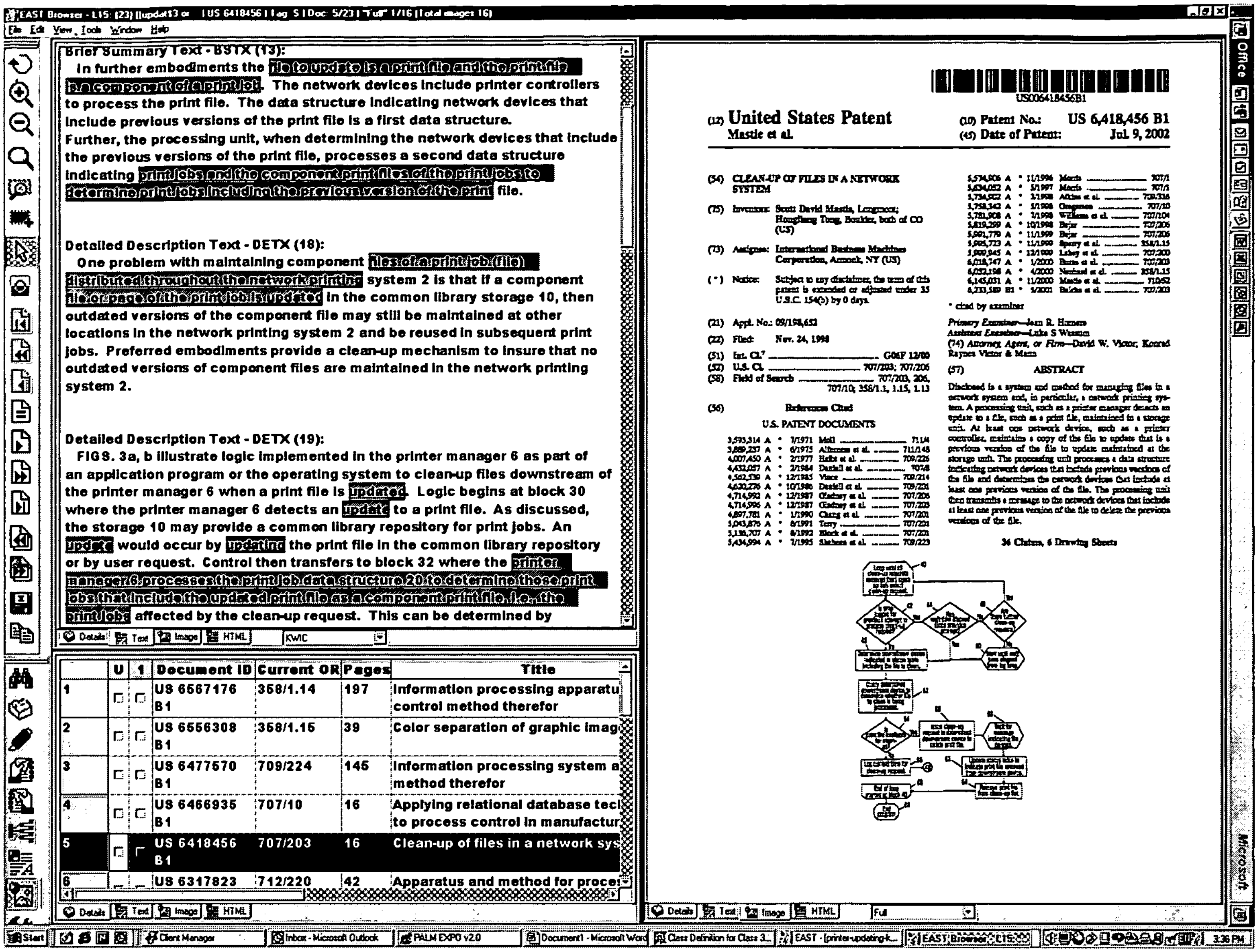
* cited by examiner

Primary Examiner—Mark Zimmerman
Assistant Examiner—Lance W. Sealey
(74) *Attorney, Agent, or Firm*—Fitzpatrick, Cella, Harper & Scinto

(57) ABSTRACT

A server machine receives a printing job, including information to be printed, and a first print parameter, and based on that information, sets a second print parameter that is suitable for the information and also alters the first print parameter. A printer prints the received information based on the printing parameters that are set or changed. Further, users who submit information that is received are identified and the printed results are stored at storage locations that differ for each user. In addition, information may be requested from an external device, and the requested information printed when it is received. Furthermore, information that is input may be transmitted to external devices to request that those devices process that information. Moreover, when an output device that is designated by a received output instruction is a locally owned apparatus, the apparatus performs the processing as instructed. When a designated output device is another device, external transmission of the output instruction is performed.

7 Claims, 156 Drawing Sheets


EAST Browser - L15: (23) ((updat13 or | US 6418456 | Tag: S | Doc: 5/23 | Format: Full 1/16 | Total images: 16)
File Edit View Tools Window Help
Details Text Image HTML KWIC
Details Text Image HTML
Details Text Image HTML Full
Start | Client Manager | Inbox - Microsoft Outlook | PALM EXPO v2.0 | Document1 - Microsoft Word | Class Definition for Class 3... | EAST - [printer-updating-k... | EAST Browser - L15... | 3:36 PM
Office
Microsoft

Brief Summary Text - BSTX (13):

In further embodiments the file to update is a print file and the print file is a component of a print job. The network devices include printer controllers to process the print file. The data structure indicating network devices that include previous versions of the print file is a first data structure. Further, the processing unit, when determining the network devices that include the previous versions of the print file, processes a second data structure indicating print jobs and the component print files of the print jobs to determine print jobs including the previous version of the print file.

Detailed Description Text - DETX (18):

One problem with maintaining component files of a print job (file) distributed throughout the network printing system 2 is that if a component file or page of the print job is updated in the common library storage 10, then outdated versions of the component file may still be maintained at other locations in the network printing system 2 and be reused in subsequent print jobs. Preferred embodiments provide a clean-up mechanism to insure that no outdated versions of component files are maintained in the network printing system 2.

Detailed Description Text - DETX (19):

FIGS. 3a, b illustrate logic implemented in the printer manager 6 as part of an application program or the operating system to clean-up files downstream of the printer manager 6 when a print file is updated. Logic begins at block 30 where the printer manager 6 detects an update to a print file. As discussed, the storage 10 may provide a common library repository for print jobs. An update would occur by updating the print file in the common library repository or by user request. Control then transfers to block 32 where the printer manager 6 processes the print job data structure 20 to determine those print jobs that include the updated print file as a component print file, i.e., the print jobs affected by the clean-up request. This can be determined by

	U	1	Document ID	Current OR	Pages	Title
1	□	□	US 6567176 B1	358/1.14	197	Information processing apparatu control method therefor
2	□	□	US 6556308 B1	358/1.16	39	Color separation of graphic imag
3	□	□	US 6477570 B1	709/224	145	Information processing system a method therefor
4	□	□	US 6466935 B1	707/10	16	Applying relational database tec to process control in manufactur
5	□	□	US 6418456 B1	707/203	16	Clean-up of files in a network sys
6			US 6317823	712/220	42	Apparatus and method for proce

US006418456B1

(12) **United States Patent**
Mastie et al.

(10) **Patent No.:** US 6,418,456 B1
(45) **Date of Patent:** Jul. 9, 2002

(54) CLEAN-UP OF FILES IN A NETWORK SYSTEM

(75) Inventors: Scott David Mastie, Longmont; Hongliang Tong, Boulder, both of CO (US)

(73) Assignee: International Business Machines Corporation, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: 09/198,652

(22) Filed: Nov. 24, 1998

(51) Int. Cl.⁷ G06F 12/00
(52) U.S. Cl. 707/203; 707/206
(58) Field of Search 707/203, 206, 707/10; 358/1.1, 1.15, 1.13

(56) References Cited

U.S. PATENT DOCUMENTS

3,593,314	A	*	7/1971	Moll	711/4
3,889,237	A	*	6/1975	Alferness et al.	711/148
4,007,450	A	*	2/1977	Haibt et al.	709/226
4,432,057	A	*	2/1984	Daniell et al.	707/8
4,562,539	A	*	12/1985	Vince	709/214
4,620,276	A	*	10/1986	Daniell et al.	709/201
4,714,992	A	*	12/1987	Gladney et al.	707/206
4,714,996	A	*	12/1987	Gladney et al.	707/203
4,897,781	A	*	1/1990	Chang et al.	707/201
5,043,876	A	*	8/1991	Terry	707/201
5,138,707	A	*	8/1992	Block et al.	707/201
5,434,994	A	*	7/1995	Shaheen et al.	709/223
5,574,906	A	*	11/1996	Morris	707/1
5,634,052	A	*	5/1997	Morris	707/1
5,734,922	A	*	3/1998	Aiken et al.	709/316
5,758,342	A	*	5/1998	Gregerson	707/10
5,781,908	A	*	7/1998	Williams et al.	707/104
5,819,259	A	*	10/1998	Bejar	707/206
5,991,779	A	*	11/1999	Bejar	707/206
5,995,723	A	*	11/1999	Sperry et al.	358/1.15
5,999,945	A	*	12/1999	Lahey et al.	707/200
6,018,747	A	*	1/2000	Burns et al.	707/203
6,052,198	A	*	4/2000	Neuhard et al.	358/1.15
6,145,031	A	*	11/2000	Mastie et al.	710/52
6,233,589	B1	*	5/2001	Balcha et al.	707/203

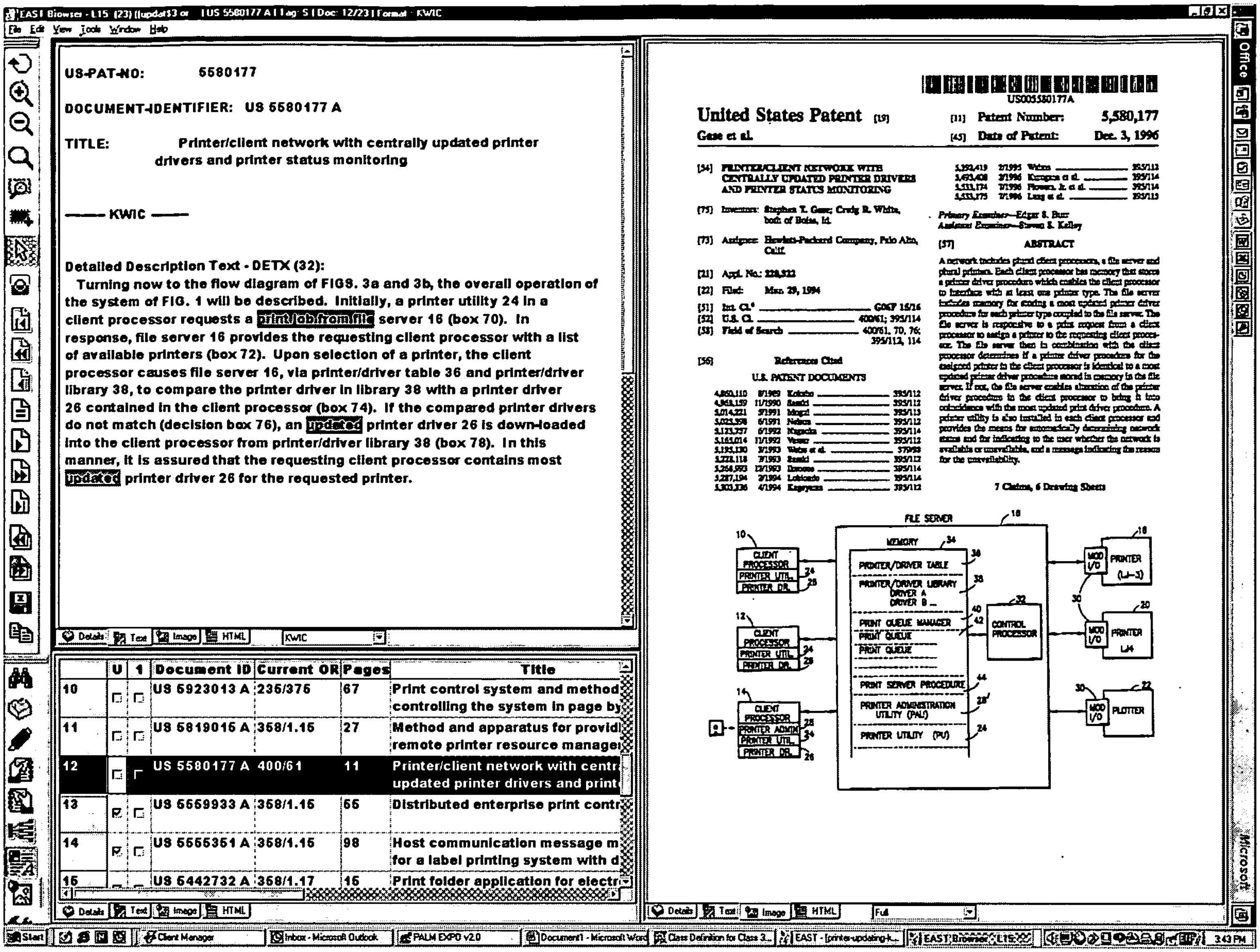
* cited by examiner

Primary Examiner—Jean R. Homere
Assistant Examiner—Luke S Wassum
(74) *Attorney, Agent, or Firm*—David W. Victor; Konrad Raynes Victor & Mann

(57) ABSTRACT

Disclosed is a system and method for managing files in a network system and, in particular, a network printing system. A processing unit, such as a printer manager detects an update to a file, such as a print file, maintained in a storage unit. At least one network device, such as a printer controller, maintains a copy of the file to update that is a previous version of the file to update maintained at the storage unit. The processing unit processes a data structure indicating network devices that include previous versions of the file and determines the network devices that include at least one previous version of the file. The processing unit then transmits a message to the network devices that include at least one previous version of the file to delete the previous versions of the file.

36 Claims, 6 Drawing Sheets



EAST Browser - L15: (23) ((updat$3 or | US 5580177 A | Tag: S | Doc: 12/23 | Format : KWIC
File Edit View Tools Window Help

US-PAT-NO: 5580177

DOCUMENT-IDENTIFIER: US 5580177 A

TITLE: Printer/client network with centrally updated printer drivers and printer status monitoring

------- KWIC -------

Detailed Description Text - DETX (32):

Turning now to the flow diagram of FIGS. 3a and 3b, the overall operation of the system of FIG. 1 will be described. Initially, a printer utility 24 in a client processor requests a print job from file server 16 (box 70). In response, file server 16 provides the requesting client processor with a list of available printers (box 72). Upon selection of a printer, the client processor causes file server 16, via printer/driver table 36 and printer/driver library 38, to compare the printer driver in library 38 with a printer driver 26 contained in the client processor (box 74). If the compared printer drivers do not match (decision box 76), an updated printer driver 26 is down-loaded into the client processor from printer/driver library 38 (box 78). In this manner, it is assured that the requesting client processor contains most updated printer driver 26 for the requested printer.

	U	1	Document ID	Current OR	Pages	Title
10	□	□	US 5923013 A	235/375	67	Print control system and method controlling the system in page by
11	□	□	US 5819015 A	358/1.15	27	Method and apparatus for providi remote printer resource manage
12	□	□	US 5580177 A	400/61	11	Printer/client network with centr updated printer drivers and print
13	☑	□	US 5559933 A	358/1.15	55	Distributed enterprise print contr
14	☑	□	US 5555351 A	358/1.15	98	Host communication message m for a label printing system with d
15			US 5442732 A	358/1.17	15	Print folder application for electr

US005580177A

United States Patent [19]

Gase et al.

[11] Patent Number: 5,580,177

[45] Date of Patent: Dec. 3, 1996

[54] PRINTER/CLIENT NETWORK WITH CENTRALLY UPDATED PRINTER DRIVERS AND PRINTER STATUS MONITORING

[75] Inventors: Stephen T. Gase; Craig R. White, both of Boise, Id.

[73] Assignee: Hewlett-Packard Company, Palo Alto, Calif.

[21] Appl. No.: 228,322

[22] Filed: Mar. 29, 1994

[51] Int. Cl.^{6} G06F 15/16

[52] U.S. Cl. 400/61; 395/114

[58] Field of Search 400/61, 70, 76; 395/112, 114

[56] References Cited

U.S. PATENT DOCUMENTS

4,860,110	8/1989	Kobobo	395/112
4,963,159	11/1990	Sasaki	395/112
5,014,221	5/1991	Mogul	395/113
5,025,398	6/1991	Nelson	395/112
5,123,757	6/1992	Nagaoka	395/114
5,165,014	11/1992	Vassar	395/112
5,195,130	3/1993	Weiss et al.	379/98
5,228,118	7/1993	Sasaki	395/112
5,268,993	12/1993	Ikenoue	395/114
5,287,194	2/1994	Lobiondo	395/114
5,303,336	4/1994	Kageyama	395/112
5,392,419	2/1995	Walton	395/112
5,493,408	2/1996	Kurogane et al.	395/114
5,533,174	7/1996	Flowers, Jr. et al.	395/114
5,533,175	7/1996	Lung et al.	395/115

Primary Examiner—Edgar S. Burr
Assistant Examiner—Steven S. Kelley

[57] ABSTRACT

A network includes plural client processors, a file server and plural printers. Each client processor has memory that stores a printer driver procedure which enables the client processor to interface with at least one printer type. The file server includes memory for storing a most updated printer driver procedure for each printer type coupled to the file server. The file server is responsive to a print request from a client processor to assign a printer to the requesting client processor. The file server then in combination with the client processor determines if a printer driver procedure for the assigned printer in the client processor is identical to a most updated printer driver procedure stored in memory in the file server. If not, the file server enables alteration of the printer driver procedure in the client processor to bring it into coincidence with the most updated print driver procedure. A printer utility is also installed in each client processor and provides the means for automatically determining network status and for indicating to the user whether the network is available or unavailable, and a message indicating the reason for the unavailability.

7 Claims, 6 Drawing Sheets

TDB-ACC-NO: NN81091893

DISCLOSURE TITLE: Checkpointing for Printer Restart (Sequence Number Method). September 1981.

PUBLICATION-DATA: IBM Technical Disclosure Bulletin, September 1981, US

VOLUME NUMBER: 24

ISSUE NUMBER: 4

PAGE NUMBER: 1893 - 1895

PUBLICATION-DATE: September 1, 1981 (19810901)

CROSS REFERENCE: 0018-8689-24-4-1893

DISCLOSURE TEXT:

3p. In computer-connected output printers, when an error is detected by the printer, the host computer will be notified by an appropriate error status indication signal. The host computer will then solicit print job restart recovery information from the printer.

Based on the information, the host processor can determine where it must resume transmission in the data stream that was sent in order to restart the job at the beginning of a new sheet of paper corresponding numerically to th page where the error ccurred. If the printer is in perative, the job may actually be restarted n another printer sinc th inf rmati n tells the h st where t begin retransmission.

The host processor insures that the first byte of data retransmitted begins at the specified Request Unit (RU) sequence number and provides to the printer the checkpoint data necessary to resynchronize to the page on which printing is to resume. N pages are bypassed without printing by the printer to avoid duplicating those pages printed without error, from the checkpoint to the page on which the error occurred or a previous page.

- Figs. 1 through 4 illustrate possible restart conditions involving data streams that may have transparent data or compressed/ compacted data within them. The recovery point is defined based on the assumption that the printer keeps a remembrance of Request Unit sequence numbers passed in the standard SNA (System Network Architecture) format headers to identify blocks of data.

- The first case is that illustrated in Fig. 1. A simple data stream containing no parameterized data is assumed. The checkpoint occurs at some arbitrary byte in the data stream every K pages at the first printed character following the page ejection, as specified by the host processor set checkpoint interval command. In order to

inf rm the host comput r where the printer was in the printing j b at the time the err r was detected, it is necessary t provide identification of the current Request Unit sequence number. This number if maintained in a register which is updated with each new Request Unit header. It is also necessary to provide the position in number of bytes which have elapsed in the job since the start of the current Request Unit.

This is defined as the control sequence offset count and is a number maintained in a counter register, that is, incremented with each byte of data printed. In Fig. 1, a checkpoint has occurred at the indicated spot and a Delta(1) exists from the start of the given Request Unit. By providing the control sequence number N and the Delta(1) offset count as a number of bytes actually printed up to the time the checkpoint occurs, the host computer will be informed of where to begin retransmission of the data. The necessary counters and logic circuitry for storing the counts are not illustrated since these are obvious to those skilled in the art or can be implemented in microcode routines.

Fig. 2 illustrates the case where transparent data occurs, but is not contained in compressed/compacted data streams. The start of transparent data can occur in one Request Unit and the checkpoint may occur in an ther Request Unit as illustrated. Sequ nce

numbers are st r d as in the previous example and updated with each new Request Unit. When transparent data occurs in the data stream, a register is stored with the location of the start of transparent data within the Request Unit and another counter is started to count the offset from the start of transparent data to the checkpoint. The counter which is counting the offset bytes from the start of transparent data will be stopped and the results stored when a checkpoint occurs. The data to be provided to the host then includes the Request Unit sequence number where the transparency data began, the Delta(1) control sequence offset where the beginning of transparent data occurred, and the Delta(2) offset occurring within the transparent data stream from the start of transparent data to the point where the checkpoint occurred.

- A third case exists as shown in Fig. 3. This case pertains to that where compressed or compacted data occurs which does not contain any standard character string control codes. In case 3, the current request response sequence number is stored in the register as with the previous two cases. When a string control byte (SCB) occurs to signal the start f c mpressed r c mpacted data, an ther register is l aded with the count of the numb r of bytes executed within the current Request Unit up to the point where the SCB was detected. Another counter is started to keep a count of the bytes elapsed during the compressed or compacted data stream up to the point where the checkpoint occurs.

The host computer must then be provided with the sequence number where the SCB occurred, the Delta(1) offset from the start of that RU to the point within the sequence number where the SCB started, and the offset Delta(2) from the SCB to the point where the checkpoint occurred.

- Fig. 4 illustrates the case case where a compressed or compacted data stream does contain standard character stream control codes. It will be been that Fig. 4 is a combination of those cases shown in Figs. 2 and 3. Current Request Unit sequence numbers are maintained in registers as before and a counter operates from the start of each sequence number until a string control byte is encountered, whereupon the Delta(1) offset from the start of the current Request Unit is stored as the starting point for the compressed or compacted data. Another counter is started at this point to measure the distance elapsed in the data stream until the b ginning of transparent data is enc unter d. This is the ffset

D lta(2) shown in Fig. 4.

This count if similarly stored In the register and an ther counter is begun measuring the offset within the transparent data portion of the data stream until the checkpoint occurs. To recover the job and begin printing at the appropriate point, the host computer must be given the sequence number of the Request Unit and the three Delta offsets, as illustrated, to locate the position within the data stream where the checkpoint occurred.

Although not illustrated, the data stream also contains sequence numbers for vertical and horizontal Request Units containing format commands and similar count offsets from the start of the specified Request Units to enable vertical and horizontal position recovery.

TDB-ACC-N : NN9310525

DISCLOSURE TITLE: Identification and Handling IPDS Resources In Error

PUBLICATION-DATA: IBM Technical Disclosure Bulletin, October 1993, US

VOLUME NUMBER: 36

ISSUE NUMBER: 10

PAGE NUMBER: 525 - 526

PUBLICATION-DATE: October 1, 1993 (19931001)

CROSS REFERENCE: 0018-8689-36-10-525

DISCLOSURE TEXT:

Many IPDS Negative Acknowledgments (NACKs) can occur either on a page or within a printer resource. Although a printer can report if one or multiple resources were involved in an error, prior to this invention this information was not actively used by IPDS print drivers to affect recovery from a particular printer problem. If it can be determined that a resource itself is the cause of a reported error, additional recovery actions are desirable to ease diagnosis and av id r p ated discovery and rep rting of the error.

- Utilizing the existing Res urce Identifi r fi lds in IPDS NACKs, the kern l resource which has caused the NACK, if any, is identified. Proper identification allows helpful reporting to the user, and modification of error recovery actions for the NACK. In abstract, the invention is to:

1. Identify the kernel resource in error, if any. If a kernel resource is identified then:
    a. Report the external name of the kernel resource to the user toaid problem resolution.
    b. Determine if the kernel resource is causal.
    c. If the resource in error is causal, change the recovery actionsfrom the NACK to cause the print job to be terminated (insteadof just the page in error, for example). This also results in achanged recovery message to the end user.

This invention is instantiated in PSF/2.

- For IPDS printers which return 24 sense-byte NACKs, information can be returned about the Overlay, Page Segment, or Font which caused (or was associated with) the error.
- Previously no use of this information was made other than to echo it to the user, and it was not considered possible and desirable to use this information to modify the user messages generated and the recovery actions for the reported problem. The difficulty in making any use of this information is that the same NACK can be reported with nothing but zeros in these fields (indicating that the NACK

ccurr d on an IPDS page), r can b rep rted with inf rmati n in any r all of th se fi lds (indicating that the NACK occurred n a page, but within a resource). Compounding the difficulty in using this information, sometimes a resource ID is provided even though the resource is not causal; ie, for some NACKs (such as off the page), one or more resource IDs may be provided, but this does not indicate that the individual resources are the cause of the error.

- This invention establishes a procedure for identifying the kernel resource which is in error, effectively reporting the resource in error information to the user, and modifying recovery actions appropriately if a causal resource has been identified.
- Identifying the kernel resource in error is necessary because overlays can imbed (use) other resources (fonts, segments, or other overlays). For example, a page segment may be loaded into the printer with an image error, and this image error could be exposed when the page segment is used on its own, or when it is used in the context of an overlay which imbeds it. The latter case is interesting, since IPDS does not govern explicitly which resource IDs need to be returned: the overlay ID, the segment ID, or both could be returned depending upon the printer microcode impl mentation of IPDS.

Th ref re the kernel identificati n part f this inventi n is pportunistic, and the m st explicit res urce ID pr vided in the NACK is taken as the kernel resource. This follows since if both an overlay ID and a Segment ID are provided, it must be the case that the segment is the problem and it happens to be within an overlay; the information that we want to relay to the user is that the segment is broken, not that the overlay is, since there is no way to fix the overlay other than by fixing the segment.

- Once a kernel resource has been identified, it is determined if the resource is causal or not. A causal resource has some problem which will cause a NACK to be reported anytime that resource is used (for example, invalid image data). A non-causal resource is a victim of circumstance, whereby the use, context, or placement of a resource is not valid, but the resource itself does not contain any errors, and could be used correctly on another page. This invention makes the causal classification of the kernel resource by heuristically treating any resource identified in the X'08' class of IPDS NACKs as non-causal (for these NACKs, the resources are often just be positioned inappropriately, and have fallen off the edge of the page). In all other cases except the X'08' class of NACKs, any identified resources are causal.
- The kernel resource is always identified to the user in a m ssage which c ntains th name of the res urce in rr r (not

just its identifier), even if it is n t a causal resource. This is a significant improvement v r th pri r art, since for the first time explicit information is given to the user about the resource associated with an error. An additional message will be provided if the resource is determined to be causal, to further clarify that the identified resource caused the problem. Note that this can be extremely important, since no job which requires this resource will print correctly until this resource is fixed.

- If the kernel resource is also causal, two things occur. First, the resource is marked as broken to prevent any subsequent use of this resource until it is fixed. This is important, since it has now been established that any subsequent use will cause the reported error to reoccur. Keeping track of the broken resource prevents other users and jobs from encountering the same broken resource until it is fixed. Secondly, the recovery actions for the print job are changed to terminate the job (instead of just the page that contained the error, for example). The rationale here is two-fold: firstly, this is a serious error that demands immediate attention, and secondly, jobs are often homogeneous, and it's likely that subsequent pages of this job will contain references to this resource and will therefore be unprintable anyway, so this avoids wasting paper or machine time.
- Note that multiple NACKs rep rt d tog ther in a single

NACK stack, whether f r the same page or for multiple pages, can each have a different kernel (and possibly causal) resource. This algorithm as instantiated within the PSF/2 produce handles any such multiple-resources-in-error case correctly too.

TDB-ACC-N : NN8707816

DISCLOSURE TITLE: Comprehensive TEST Tool - an Integrated Testing
Development Tool

PUBLICATION-DATA: IBM Technical Disclosure Bulletin, July 1987, US

VOLUME NUMBER: 30

ISSUE NUMBER: 2

PAGE NUMBER: 816 - 821

PUBLICATION-DATE: July 1, 1987 (19870701)

CROSS REFERENCE: 0018-8689-30-2-816

DISCLOSURE TEXT:

- Comprehensive Test Tool (CTT) is an integrated testing development tool that coordinates and standardizes testing for developers, testers, managers and system assurance auditors. CTT can be used not only for component testing, but also for build verification, development, product and system testing. CTT is a menu-driven tool executed on a virtual machine operating system (VM) with SQL (Sequential Queries Language) data base support. The data base contains department information, component information and test

inf rmati n. The latt r includes inf rmati n n variati ns, test cases, test pr grams, t st plans, test buck ts and test status.
*****
SEE ORIGINAL DOCUMENT ***** Using CTT, testers and developers can produce reports with this information for themselves, their managers and system assurance personnel.

Any testing-related documents, test programs and status reports can be generated by using the information in CTT's central data base. Fig. 1 illustrates the major CTT functions which support testing development. The Native system (the system being built) is used to compile the Native High Level Language programs and execute the Test Cases. A communication line is needed between CTT and the Native System in order to download or upload the testing information. The master copy of that information (which includes Test Plan documentation, Test Case descriptions, Variation descriptions, Test Program code, Test Buckets and Test Results) is stored in the VM system data base. The Comprehensive Test Tool complies with the testing development process. CTT provides the user with six major functions. These functions are listed below.
*****
SEE ORIGINAL DOCUMENT ***** 1.Integrated Test 1 (IT1) Package Development This function:

C mp nents, High Level IDs, L w Level IDs and

Keyw rds.

- br ws and print Variati ns f r a Component.
- br ws and print Test Cas s. 2.Integrated T st 2 (IT2)

Package Development This function: ;. ALLOWS THE USER TO CREATE,

UPDATE, DELETE, COPY, RENAME,

compile/bind, browse and print Test Programs for a Component. DATA. AND

browse an Include file. 3.Test Plan Development This function: IT1

or IT2 into a Test Plan for review. PRINT

request. 4.Test Bucket Generation. A Test Bucket is a single program that

will execute all of the Test Cases it contains. This function: ON

one or more of the following: Component, High Level ID, Low

Level ID, Keyword, Test Case Status.

- compile/bind the Test Bucket. up date the Master

Component status. 5.Queries/Reports This function allows the user to

ask about test information. The

user may present that information on-line or print it. Test information includes Keywords, Variations, Test Programs, Test

Cases, Components, Test Buckets and Departments. 6.User Profile

Maintenance This function:

. allows the user to change the user name, department, upline

department and preferred editor fields in his or her CTT user

profile. The Comprehensive Test Tool has the following features: T Coordinates the Testing Process CTT allows the user to

d vel p Test Plans, cod Test Pr grams,

create Test Buckets and g nerate Test Results. These CTT func
ti ns are executed on VM with SQL data base support.
T Coordinates
the Sharing of Data Test information is centrally stored in the SQL
data base.
Centrally stored information allows the user to easily view and
efficiently copy another CTT user's test data. CTT provides
skeleton files for Test Programs and Test Plans.
Information entered into CTT by the user is embedded into these
skeleton files. This improves the user's speed and accuracy.
GENERATION The S-Curve and Test Matrices are generated and embedded
into the
IT1 Package. CTT generates the Test Bucket Driver Program which
evokes associated Test programs. T STANDARDIZES TEST RESULTS AND
REPORTS After a Test Bucket is created on VM, it is sent to the
Simulator
or to the Native System to be tested. Test Results are recorded
in a standard format and returned to VM.
A Test Results Report,
also in a standard format, can be generated for browsing or
print ing.
.SECURITY CHECKING Only the owner of a Component can use the READ
and WRITE functions
for that Component; other CTT users can only view or copy the

test
inf rmati n for that Component.
.CONSISTENT FUNCTI N KEYS Functi n keys are c nsistent
from screen
to screen.
.HELP TEXT FOR EACH SCREEN
Each screen has an on-line Help text which provides
detailed
information relating to that screen.
.COMPILE/BIND FUNCTION CTT interfaces with IDSS to
access the proper
compiled/bind screen
for the user who wishes to use the compile/bind function.
.PRINTING CAPABILITY The user can print any CTT test
information
using the CTT print
function.
The test information includes Variations, Test Cases,
Test Programs, Test Buckets, Test Plans and Test Reports.
.BATCH JOB SUBMISSION For the print and compile/bind
functions, CTT
allows the user to
submit the job through immediate or overnight batch job
submis
sion.
.LIST PROCESSING
The user may ask CTT to search its data base and display a
list
for a specific input field. The user may leave an input field
blank to receive the complete list of valid choices. Or the
user
may complete part of the input field so that CTT will limit
the
list to the user's input field specifications.

SECURITY: Use, c pying and distributi n f this data is subject

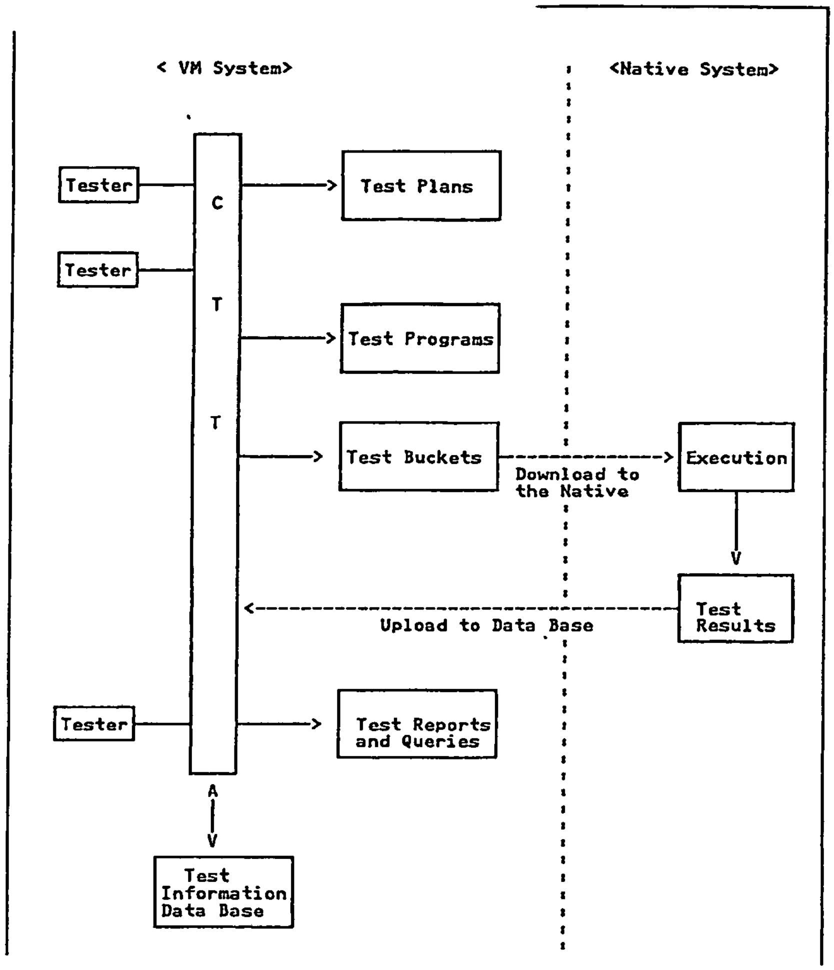
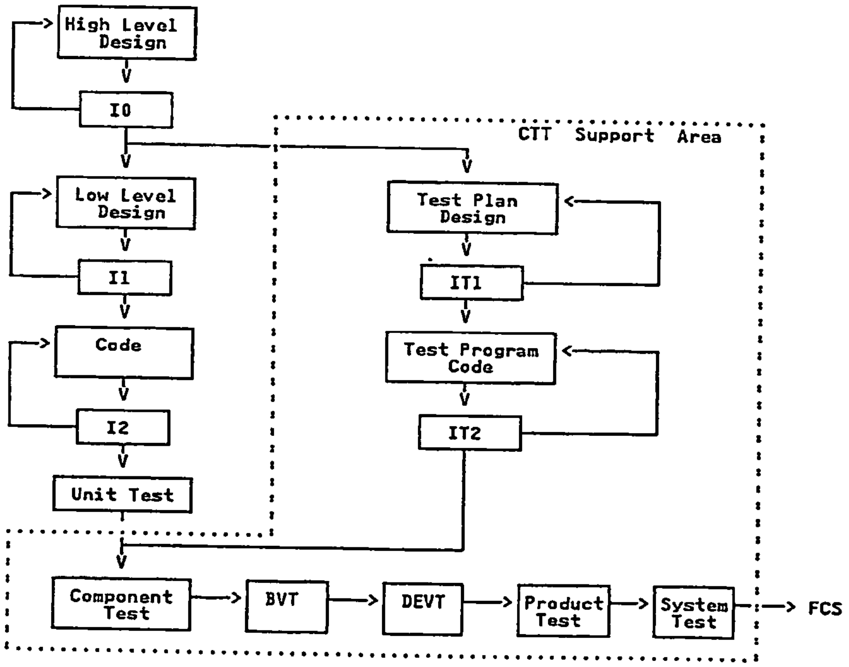


Fig. 1



Note: BVT - Build Verification Test
DEVT- Development Test
FCS - First Customer Ship

Fig. 2

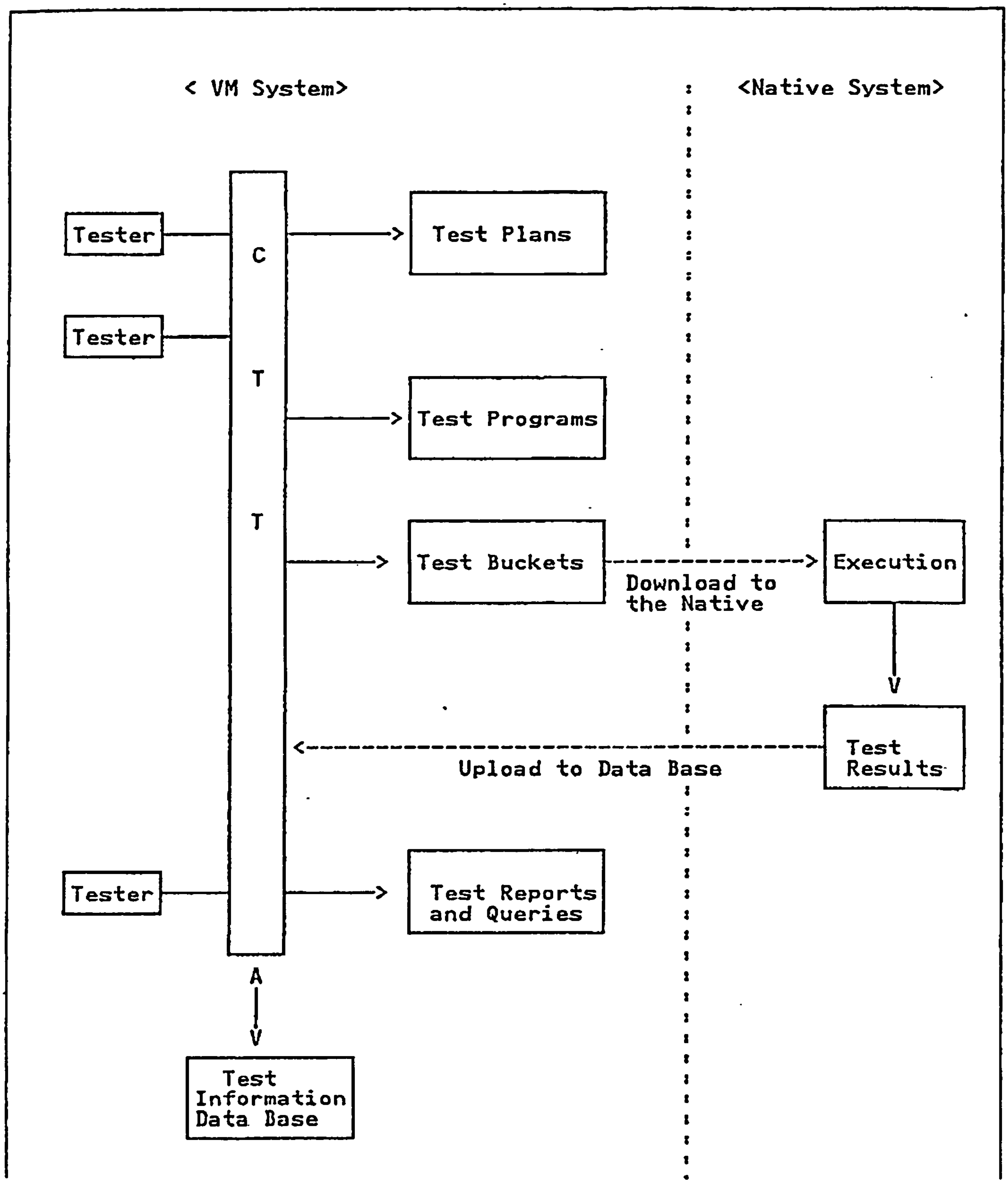

< VM System>
<Native System>
Tester
Tester
Tester
C
T
T
Test Plans
Test Programs
Test Buckets
Download to the Native
Execution
V
Test Results
Upload to Data Base
Test Reports and Queries
A
V
Test Information Data Base

Fig. 3

WEST Search History

```
DATE:  Wednesday, August 13, 2003
```

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=ADJ			
L8	l3 and ((updat$3 or install$3 or upgrad$3 or modif$6) with printer$2)	21	L8
L7	l4 and ((updat$3 or install$3 or upgrad$3 or modif$6) with printer$2)	0	L7
DB=TDBD; PLUR=YES; OP=ADJ			
L6	Method with (Updating Microcode) with (Peripheral System).ti.	1	L6
L5	"Method of Updating Microcode in a Peripheral System".ti.	0	L5
DB=USPT; PLUR=YES; OP=ADJ			
L4	L3 and ((717/168 |717/169 |717/170 |717/171 |717/172 |717/173 |717/174 |717/175 |717/176 |717/177 |717/178)!.CCLS.)	14	L4
L3	(updat$3 or install$3 or upgrad$3 or modif$6) with microcode$2	643	L3
L2	(print$3 with (updat$3 or install$3 or upgrad$3 or modif$6)) with microcode$2	3	L2
L1	(print$3 with job$2) with microcode$2	3	L1

```
END OF SEARCH HISTORY
```

	Type	Hits	Search Text	DBs
1	BRS	25	(print adj job$1) and microcode	USPAT; US-PGPUB
2	BRS	1421	(print adj job$1) and updat$6	USPAT; US-PGPUB
3	BRS	165	(print adj job$1) with updat$6	USPAT; US-PGPUB
4	BRS	47	(print adj job$1) with embed$6	USPAT; US-PGPUB
5	BRS	13	(print adj job$1) and 717/168-178.ccls.	USPAT; US-PGPUB